

South Dakota State University  
**Open PRAIRIE: Open Public Research Access Institutional  
Repository and Information Exchange**

---

Electronic Theses and Dissertations


---

2017

# A Dynamic Scaling Methodology for Improving Performance of Big Data Systems

Nashmiah Alhamdawi  
*South Dakota State University*

Follow this and additional works at: <https://openprairie.sdstate.edu/etd>

 Part of the [Data Storage Systems Commons](#), [Digital Communications and Networking Commons](#), and the [Electrical and Computer Engineering Commons](#)

---

## Recommended Citation

Alhamdawi, Nashmiah, "A Dynamic Scaling Methodology for Improving Performance of Big Data Systems" (2017). *Electronic Theses and Dissertations*. 2261.  
<https://openprairie.sdstate.edu/etd/2261>

This Thesis - Open Access is brought to you for free and open access by Open PRAIRIE: Open Public Research Access Institutional Repository and Information Exchange. It has been accepted for inclusion in Electronic Theses and Dissertations by an authorized administrator of Open PRAIRIE: Open Public Research Access Institutional Repository and Information Exchange. For more information, please contact [michael.biondo@sdstate.edu](mailto:michael.biondo@sdstate.edu).

A DYNAMIC SCALING METHODOLOGY FOR IMPROVING PERFORMANCE OF  
BIG DATA SYSTEMS

BY

NASHMIAH ALHAMDAWI

A thesis submitted in partial fulfillment of the requirements for the

Master of Science

Major in Computer Science


South Dakota State University

2018

A DYNAMIC SCALING METHODOLOGY FOR IMPROVING PERFORMANCE OF  
BIG DATA SYSTEMS

NASHMIAH ALHAMDAWI


This thesis is approved as a creditable and independent investigation by a candidate for the Master of Science in Computer Science degree and is acceptable for meeting the thesis requirements for this degree. Acceptance of this does not imply that the conclusions reached by the candidates are necessarily the conclusions of the major department.

 Liu, Ph.D.  
Thesis Advisor

Date

Steven Hietpas, Ph.D.  
Department Head  
Dept. of Electrical Engineering and Computer Science

Date

 Dean, Graduate School

Date

## ACKNOWLEDGMENTS

“I've learned in my life that it's important to be able to step outside your comfort zone and be challenged with something you're not familiar or accustomed to. That challenge will allow you to see what you can do.”

J. R. Martinez

Without a doubt, I was challenged and put out of my comfort zone, but thanks to magnificent people that helped me in this interesting journey I have encountered, I am able to say that I am a richer person. Not only did I learn that I can overcome and succeed what I put my mind into, but I also learned that there is a lot of people that are willing to go out of their way to help me, and see me succeed. Not only did I overcome challenges throughout this whole process, but I did so successfully.

I would like to give a special thanks to Dr. Yi Liu for helping me and guiding me throughout the entire process. Without your knowledge and guidance, I would not stand where I do today. Thanks to your support I can now see what I am truly capable of; I know that if I push hard enough I am able to do whatever I set my mind too.

They say good friends are hard to find, but extraordinary friends are like hidden treasure that only few are able to obtain. Extraordinary friends want to see you succeed and will help you with what it ever it is in order to obtain that success. I can honestly say that I have that these last months have seen the love that my friends had towards me not only did they encourage me to keep going, but helped me throughout the process. Thank you all for cheering me on and helping me achieve my goals.

But far beyond anything thank you to my family to whom without I would not be here at all. There are no words to describe my gratitude that I have towards those who have shown unconditional love and support for every decision I have made. Not only did they tell me to keep going whenever I had lost hope, but have also left their comfort zone to support my decision to be here. Again, no words can describe my gratitude towards the abundance of love they had shown.

And now I know what I am capable of that even in my lowest I can get up, and I have people that will help me rise up and keep on going to the goal, thank you so much to everybody that have helped me reach this amazing goal.

## CONTENTS

LIST OF FIGURES .....	vii
LIST OF TABLES .....	viii
ABSTRACT.....	ix
Chapter 1 INTRODUCTION.....	1
1.1 Introduction.....	1
1.2 Motivation.....	2
1.3 Objective .....	2
1.4 Thesis Organization .....	3
Chapter 2 BACKGROUND.....	4
2.1 EASTWeb V2.3 .....	4
2.2 Cloud Computing .....	7
2.2.1 Elastic Computer Cloud Service (Amazon EC2).....	7
2.2.2 Cloud Storage Service (Amazon S3) .....	8
2.2.3 Amazon Relational Database Service (Amazon RDS).....	8
2.2.4 AWS CodeDeploy Service.....	9
2.3 Related Work .....	10
2.3.1 Dynamic Scalability of Web Applications on a Virtualized Cloud Computing Environment.....	10
2.3.2 ECG Monitoring and Analysis System.....	11
2.3.3 High Performance Analytics of Big Data with Dynamic and Optimized	

Hadoop Cluster .....	13
Chapter 3 METHODOLOGY .....	17
3.1 Scaling the System .....	17
3.1.1 Helper Project Algorithm .....	17
3.1.1.1 Consider Categories of Data in Splitting .....	18
3.1.1.2 Consider Analyzing Data in Splitting .....	19
3.1.1.3 Consider Volume of Data in Splitting .....	19
3.2 Modify the Current System .....	20
3.3 Deployment Environment .....	21
3.4 Database Transformation .....	23
3.5 Guidelines .....	25
Chapter 4 CASE STUDY .....	28
4.1 EASTWeb V2.3 Overview .....	28
4.2 Amazon Web Service (AWS) .....	29
4.3 How to Apply Methodology .....	29
4.3.1 Helper Project Algorithm .....	30
4.3.1.1 Rule 1: Number of Plugins Considered as Number of Categories .....	30
4.3.1.2 Rule 2: Number of Indices Considered as Number of Operations.....	31

4.3.1.3 Rule 3: Number of Years Considered as Size of Data.....	32
4.3.2 Modify the Current System .....	34
4.3.2.1 Base Version .....	34
4.3.2.2 Virtual Machine Version.....	36
4.3.3 Database Transformation .....	38
Chapter 5 EVALUATION.....	41
5.1 How Does the Dynamic Scaling Methodology Work? .....	41
5.1.1 Applying Dynamic Scaling Methodology in the Case Study .....	41
5.1.2 Performance Comparison of Case Study .....	43
5.2 How Does Deployment Approach Work? .....	45
5.2.1 Simplicity.....	45
5.2.2 Flexibility .....	46
5.3 Guideline .....	47
5.4 Compare our Work with other Works .....	47
5.4.1 Dynamic Scalability of Web Applications on a Virtualized Cloud Computing Environment.....	47
5.4.2 ECG Monitoring and Analysis System.....	48
5.4.3 High Performance Analytics of Big Data with Dynamic and Optimized Hadoop Cluster .....	48
Chapter 6 CONCLUSION .....	49



6.1 Conclusion .....	49
6.2 Future Work.....	50
6.2.1 Consider other Deployment Environment .....	50
6.2.2 Consider the Vertical Scaling .....	51
LITERATURE CITED .....	52

## LIST OF FIGURES

Figure 1. The Major Processing Steps in EASTWeb System.....	5
Figure 2. The High-level Architecture of the EASTWeb .....	6
Figure 3. The Workflow of AWS CodeDeploy Service .....	9
Figure 4. Architecture to Scale Web Applications in a Cloud.....	11
Figure 5. The System Design of ECG Components of Personal Health Monitoring System .....	13
Figure 6. The System Components .....	15
Figure 7. Dynamic Scaling Methodology Steps and Requirements .....	24
Figure 8. Overview of Dynamic Scaling Methodology .....	30
Figure 9. The Workflow of Two Versions of EASTWeb.....	39

## LIST OF TABLES

Table 1. Functions of Helper Algorithm.....	34
Table 2. The Performance Results of Two EASTWeb Scenarios .....	44

## ABSTRACT

A DYNAMIC SCALING METHODOLOGY FOR IMPROVING PERFORMANCE OF  
BIG DATA SYSTEMS

NASHMIAH ALHAMDAWI

2018

The continuous growth of data volume in various fields such as, healthcare, sciences, economics, and business has caused an overwhelming flow of data in the last decade. The overwhelming flow of data has raised challenges in processing, analyzing, and storing data, which lead many systems to face an issue in performance. Poor performance of systems creates negative impact such as delays, unprocessed data, and increasing response time.

Processing huge amounts of data demands a powerful computational infrastructure to ensure that data processing and analysis success [7]. However, the architectures of these systems are not suitable to process that quantity of data. This calls for necessity to develop a methodology to improve the performance of systems handle massive amount of data.

This thesis presents a novel dynamic scaling methodology to improve the performance of big data systems. The dynamic scaling methodology is developed to scale up the system based on the several aspects from the big data perspective. Moreover, these aspects are used by the helper project algorithm which is designed to divide a task into small chunks to be processed by the system. These small chunks run on several virtual machines to work in parallel to enhance the system's runtime performance. In addition,

the dynamic scaling methodology does not require many modifications on the applied, which makes it easy to use.

The dynamic scaling methodology improves the performance of the big data system significantly. As a result, it provides a solution for performance failures in systems that process huge amount of data. This is study would be beneficial to IT researches that focus on performance of big data systems.

## Chapter 1 Introduction

### 1.1 Introduction

Currently we live in an age where big data has emerged and is drawing attention in several fields such as science, healthcare, business, finance, and society. The continuous growth in data size in various fields has caused an overwhelming flow of data in the last decade. Thus, many systems have faced problems analyzing, storing, and processing large quantities of data, which in turn has caused performance failures or slower performance and processing. When experiencing poor performance in systems processing huge amounts of data, a negative impact is created in increased costs, decreased revenue, or both. Additionally, poor performance causes delays, creates unprocessed data, and increases response time. The question is: Does handling big amounts of data play a significant role or have a great impact in performance? Also, how do we improve the performance of systems when dealing with huge amounts of data?

Under the exponential growth of data, the term big data refers to the increase in the volume of data and difficulty to store, process, and analyze through traditional databases [7]. Furthermore, it is characterized in the 4Vs: variety, value, velocity, and volume.

Big data raises many challenges in performance, scalability, and capacity. With big data systems becoming more prevalent, a need exists to overcome the challenges and implications of massive data. Therefore, this thesis aims to illustrate how to improve performance in big data systems, and it focuses on the performance challenge from the perspective of big data systems. The problem tackled in this thesis is performance failures

in systems that process and analyze enormous amount of data. Also, many systems deal with huge datasets and request rates beyond the capabilities of relational databases and performance systems. Thus, a novel methodology is illustrated to dynamically scale up these systems.

## **1.2 Motivation**

Data growth is rapidly outpacing the capability of systems to store, process, and analyze the data that they are collecting, which then affect the performance of these systems [15]. The contributions of this thesis discuss processing big data in a timely manner, dynamic scaling up of systems based on several rules, and flexibility of using a deployment environment. In addition, this thesis would be beneficial to information technology researchers who focus on improving performance and scalability of big data systems.

There are many big data systems face a problem with unprocessed data because of the size of these data. For example, EASTWeb system is not able to process a NLDAS Forcing project in a timely manner. This project processes huge amount of data, such as processing 39 years of data requires 11.735 TB. Thus, that causes poor performance of these systems and then creates unprocessed data.

## **1.3 Objective**

This thesis aims to identify how to enhance the performance of big data systems. The objectives of this study are to develop a dynamic scaling methodology using virtualization to improve performance and to establish an approach to adapt a system to

the deployment environment. Moreover, this thesis seeks to determine guidelines for systems facing the same issues in processing huge amount of data.

#### **1.4. Thesis organization**

This thesis presents a novel dynamic scaling methodology to improve performance by applying an algorithm to scatter the system and scale up/down the system in the deployment environment. It provides solutions for problems related to scalability and distributed systems. Moreover, it describes how to utilize the deployment environment along with scalability of the system. The outline of the remainder of the thesis is organized as follow. Chapter 2 provides an overview of the methods used to apply the approach, such as EASTWeb and Cloud computing, and discusses related works similar to our approach. Chapter 3 describes the structure of methodology to scatter and scale up/down big data systems; it also shows the helper project algorithm based on three rules and brief guidelines of the dynamic scaling approach. Chapter 4 depicts how to implement a helper project algorithm at EASTWeb as a case study and an overview of EASTWeb after performing the dynamic scaling methodology. Chapter 5 shows experimental results after applying the dynamic scaling methodology and comparison between EASTWeb with and without the helper project algorithm. Finally, Chapter 6 summarizes the thesis and presents suggestions for future work ideas.



## Chapter 2 Background

This chapter illustrates the concepts of the cloud computing, the overview of the EASTWeb system that is used as case study to apply dynamic scalable methodology, and the related works.

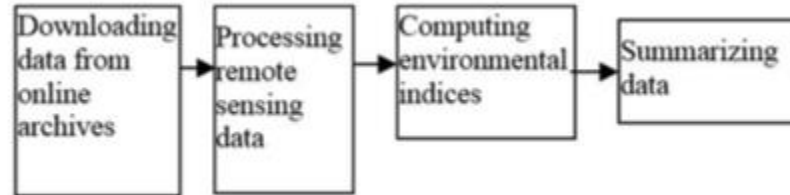
### 2.1 EASTWeb Version 2.3 [1,2]

The Epidemiological Applications of Spatial Technologies (EASTWEB) application is developed as an open-source, client-based application that automatically connects to earth science data archives. It helps acquire, process, and summarize remote sensing data according to the time period and geographic information provided by the user. All the information summarized is saved in a database that can easily inquire and connect to the data server, ecological and epidemiological, for further analyzation and prediction in a software environment.

EASTWeb is developed to facilitate the access of earth observation data for public health research and its applications. The earth science data is already maintained from various organizations, so the data structure, access methods, and file formats are different from each other. Therefore, EASTWeb is able to process different types of data products, and merge the remotely-sensed environmental data in public health research.

The Geospatial Data Abstraction Library (GDAL) is used for spatial analysis. EASTWeb is implemented in JAVA, and utilized PostgreSQL to save and manipulate the resulting of data summaries. The system inputs are a collection of earth observation data files which are already downloaded from online archives. The system outputs tables including data on the summary statistics of environment indices for each special zone.

As illustrated in Figure 1, EASTWeb includes four major processing steps which are (1) downloading data from online earth observation data archives, (2) processing remote sensing data, (3) computing environment indices, and (4) summarizing data using the zonal statistics technique.



*Figure 1. The major processing steps in EASTWeb system*

The downloading step accesses data stream files from earth observation archives and stores them locally for further use. The processing remote sensing data step includes seven sub-steps such as mosaicking, converting, compositing, data filtering, water masking, clipping and reprojecting. At the end of the processing step, the resulting data layers are stored in the Geo Tiff format. The computing environmental indices step calculates different indices for each data product, for example, NDVI, EVI, SAVI, and NDWI for MODIS NBAR. Step Four: The summarizing data step generates spatial summaries of the environmental indices based on the zonal shapefiles that the user provides and stores the results in the database.

As shown in Figure 2 the architecture of EASTWeb system contains four components designed to map the four major processing steps. The Downloading component is responsible of downloading and storing data locally. The RS Data Processing component wraps these subcomponents designing for the sub-steps in the

Processing step. The Environmental Indices component generates GeoTiff files and calculates environmental metrics. The Summarization component computes all the summaries and stores the results into database via Database Wrapper.

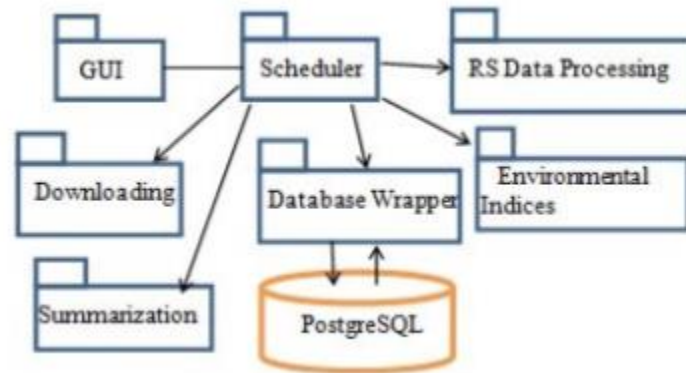


Figure 2. The high-level architecture of the EASTWeb

EASTWeb uses the Graphical User Interface (GUI) to support the user interaction with the system. The scheduler component is designed to collaborate all EASTWeb components to work together. EASTWeb implements 8 plugins for 8 different earth observation data sets including GPM's IMERG, IMERG\_RT, MODIS MOD11A2 C6, MODIS MCD43A4 C6, TRMM\_3B42, TRMM\_3B42RT, NLDAS forcing, NLDAS NOAH. Each plugin has various indices ranging from one to fifteen indices. The indices do various environmental calculations for each data product. Moreover, the user can use EASTWeb to create a project that may include several plugins and choose indices associated with the selected plugins.

## **2.2 Cloud Computing**

Cloud computing is an emerging area, supporting various fields of computing, and has become a strong architecture to apply massive-scale and complex computing. Cloud computing facilitates in providing the virtualized resource, parallel processing, data storage, and security [7]. It is invented to enable a capable access to share resources, such as computer networks, servers, storage, and application services. Moreover, it assures to be less expensive compared to supercomputers and specialized clusters, is a more reliable platform alternative to grid, and is more scalable than clusters [11].

The cloud computing environment is provided by vendors such as IBM [23], Microsoft Azure [25], Google Cloud platform [24], and Amazon Web Service [8]. Amazon Web Services (AWS) is utilized as the deployment environment of this study. AWS is a secure cloud service platform offering several functionalities helping businesses scale and grow. Several services have been used to provide the necessary resources to achieve our dynamic scaling methodology, including Elastic compute cloud service, cloud storage service, CodeDeploy service, and Relational database service.

### **2.2.1 Elastic Compute Cloud Service (Amazon EC2)**

Amazon Elastic Compute Cloud Service (Amazon EC2) is a web-based service allowing businesses to execute applications in the public cloud. Amazon EC2 allows a developer to create virtual machines (VM) known as instances, which can easily configure the capacity scaling of computing. Moreover, utilizing Amazon EC2 eliminates the needs of expensive hardware, so it provides the ability to develop and deploy applications faster [8]. Several features attract developers to Amazon EC2 for cloud computing. The most significant among them are [12]:

- Integration: EC2 is able to interact with other AWS services such as RDS and S3.
- Accurate control: users can access and control the instances such as start, stop, and boot.
- Security: users can manage the privacy of instances.
- Cost: EC2 provides reasonable hourly rates.

### **2.2.2 Cloud Storage Service (Amazon S3)**

Amazon Simple Storage Service (Amazon S3) [12] is a scalable, low-latency, affordable, web-based cloud storage service. Amazon S3 facilitates online backup, archiving data, and storing applications. Also, it is designed to make web-scale computing easier for developers. With Amazon S3, data are stored in sealable containers known as buckets. A bucket is able to store several kinds of data and can be controlled and managed by the user.

### **2.2.3 Amazon Relational Database Service (Amazon RDS)**

The Amazon Relational Database Service (Amazon RDS) is a web-based service that facilitates setting up, operating, and scaling a relational database in the cloud [8]. Amazon RDS offers an affordable, scaling capacity for an industry standard relational database. It is designed to manage database tasks such as backup, migration, and patching. Moreover, Amazon RDS supports six familiar database engines, including PostgreSQL, MySQL, Oracle, Amazon Aurora, MariaDB, and Microsoft SQL Server [8].

### 2.2.4 AWS CodeDeploy Service

AWS CodeDeploy is a deployment service that automates an application to an Amazon EC2 instance and on-premise server. For successful deployment, a developer should define three criteria's: Revision, deployment group, and deployment configuration [14]. The revision is the content deployed onto instances such as code, web, and configuration file. Also, it is stored in S3, GitHub repositories, or Bitbucket repositories to be able to be deployed by AWS CodeDeploy. In the deployment group, a set of instances related to certain applications is specified by the developers. Deployment configuration determines the steps of the deployment process to assure the revision is set at appropriate instances. Figure 3 describes the workflow of the AWS CodeDeploy service [8].

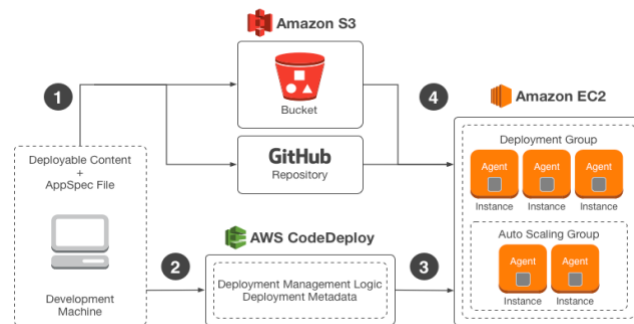


Figure 3. The Workflow of AWS CodeDeploy Service

## **2.3 Related works**

### **2.3.1 Dynamic Scalability of web applications on a virtualized Cloud**

#### **Computing environment [10]**

This research shows a scaling approach to address dynamic scaling of web applications in the cloud. The system is structured with a front-end load balancer, many web applications on virtual machines, a provisioning subsystem, and a service monitor subsystem with a dynamic scaling algorithm as shown in Figure 4. The front-end load balancer is an Apache HTTP load balancer. It is responsible for routing and balancing the user requests to the web applications published on virtual machines in the cloud, which allows the system to dynamically increase the number of servers. The service monitor subsystem gathers the number of active sessions, which is used as a scaling indicator from the web application and calculates the moving average. The scaling algorithm controls and applies the scale-up or scale-down in the provisioning subsystem on several virtual machines based on the moving average. The provisioning subsystem is responsible for deploying and cloning the virtual image to a new virtual machine. These components work together to address the dynamic scalability of the web application on the cloud environment.

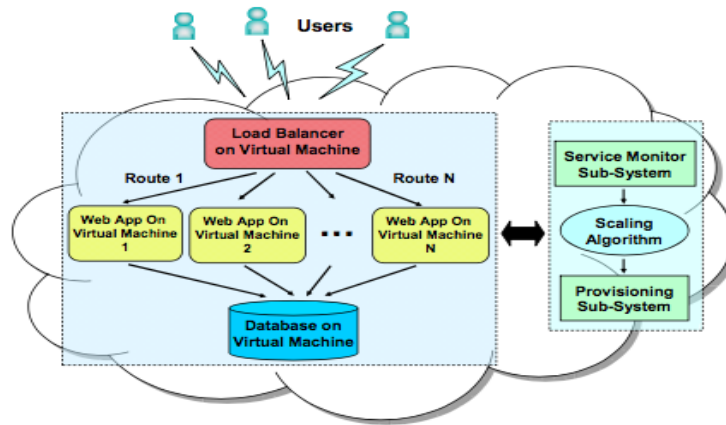


Figure 4. Architecture to scale web applications in a cloud

### 2.3.2 ECG monitoring and analysis system [9]

This research work develops a scalable health monitoring and analyzing system for people who need periodic monitoring of their health. Also, it suggests a general structure based on cloud computing to be suitable for various cases when patients need to frequently check their health; registered data must be processed to be available for specialists or patients. They use a case study to apply the approach and focus on patients who have cardiac arrhythmias. The electrocardiogram (ECG) monitoring and analysis system is developed as a case study in this paper. The ECG has several functionalities with the following steps:

1. ECG data is obtained from an ECG sensor attached on the patient and on a mobile device.
2. Data are sent to ECG analysis which is hosted via the cloud.



3. ECG analysis implements many computations and includes comparison, classification, and systematic diagnosis of heartbeats, which causes delays when performed for a large number of users.
4. Software adds the latest results of the patient's record in the cloud storage, and specialists can then later explain the features extracted from the ECG data.
5. Monitoring and analyzing processes are iterative based on user preference.

Figure 5 illustrates the system design of the ECG monitoring and analysis system and has three layers; each layer has its function and its components. The first layer is responsible of hosting the software as a web-service which is available for all users to customize the analysis of historic and current ECG data and then upload the data. Cloud computing enables the hosting of this software as SaaS. The second layer is PaaS which manages the execution based on three components: container scaling manager, workflow engine, and Aneka. The container scaling manager determines how to increase the number of containers to divide the user requests into workflow engines. This component is known as load balancing that is performed at a run time based on the average requests that the container processes in a specific time and the number of requests waiting to be scheduled in the workflow engine. The workflow engine is hosted on containers. It controls the execution of tasks of ECG data analysis when the number of user requests are increased. It packages the tasks of ECG analysis demanded by the user and transfers them to Aneka. Aneka is a platform for deploying and managing applications in Microsoft.NET framework environments. It controls communication between two different layers such as IaaS and PaaS. As part of Aneka, the dynamic scalable runtime

(DSR) is implemented to preserve the quality of service (QoS) of application running in the first layer of SaaS. It considers response time as a parameter of QoS, so DSR monitors the response time of applications and decides whether to increase the number of virtual machines.

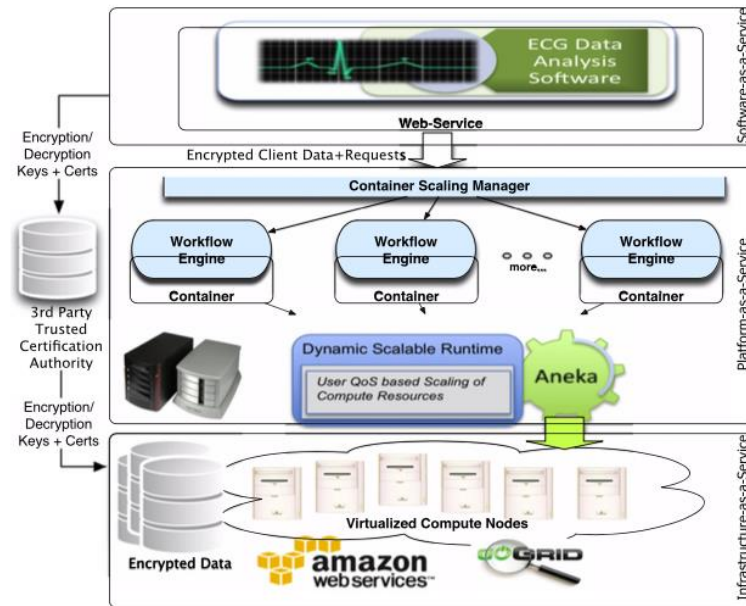


Figure 5. The system design of ECG components of personal health monitoring system

### 2.3.3 High performance analytics of big data with dynamic and optimized Hadoop cluster [16].

This research work represents an overall idea about the barriers users previously experienced when analyzing a large amount of data with essential features such as ease of accessibility, fast performance, durability, and security. It maintains users need to use a cloud-based web application that stores data in Amazon S3. Through this system, users do not need to calculate the number of nodes because the system supports a dynamic and

optimized cluster node size. Moreover, the system includes Amazon EMR and MapReduce paradigm analysis of big data analysis in the desired time.

To begin, traditional analytical tools previously presented data by processing information that was overwhelmed by the amount of collected data. When the amount of data increases and added to the workload of the platforms, several challenges show up such as storing, effective analyzing, managing, and procuring quick and correct results from data that was performed in a distributed environment. Currently available tools are the security efficient SPSS, SAS, Minitab, and R. However, two problems occur: they rely on the main memory, and they require functioning in moderate sized data sets.

Amazon EMR is a service of Amazon web services to create clusters. Amazon EMR is a Hadoop cluster which runs a MapReduce program.

The system components shown in figure 6. are described as below:

1. The Hadoop Distributed File System HDFS is a file system that allows users to use scalable and reliable data storage. It is Java based and initiated to span large clusters of commodity servers. HDFS is highly fault-tolerant, giving access to large amounts of data, and uses stream access to file system data.
2. MapReduce is a distributed processing architecture, popular programming model, and indispensable contrivance to process and generate large data sets.
3. Amazon Elastic MapReduce (Amazon EMR) is used to establish Hadoop cluster, perform and conclude designed analytic tasks, and transfer data between EC2(VM) and S3(Object Storage).
4. R programming language and Rhadoop is an open source statistical programming language containing four packages: Plymr, Rmr, Rhdfs, and Rhbase.

5. Amazon Simple Storage Service (Amazon S3) is storage for the Internet and has interfaces to save and retrieve data of desired quantity at anytime and anywhere.
6. Linear regression analysis is the most widely used statistical techniques.
7. Cluster optimization: Hadoop is a tool handling thousands of nodes and prototypes of data. To maintain the importance of optimized performances, cluster optimization is a technique to initiate the optimal number of nodes in the Hadoop cluster to apply on data.

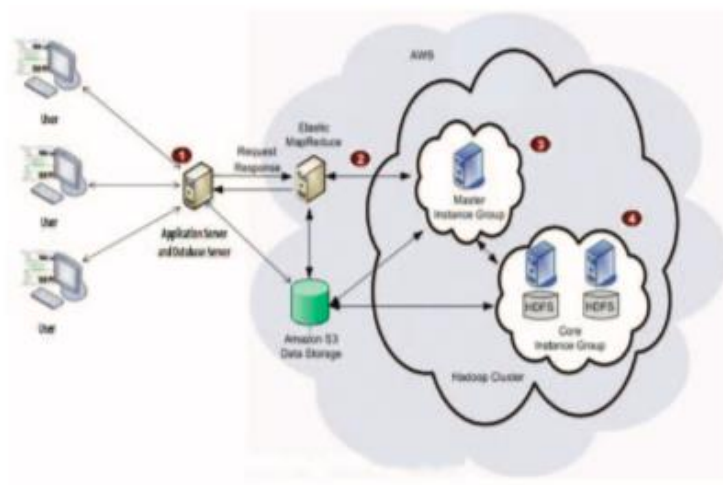


Figure 6. The system components

The system model consists of four layers:

1. The service layer keeps detailed user records allowing uploading data with no limit on time and storage and meeting the requirements of scalability, reliability, speed, and low cost.
2. The data processing layer is an open source as R integrates with Hadoop. It is used to utilize the features of the Hadoop distributed file system and MapReduce paradigm.

3. The virtualization layer is a cloud platform, an application server, and database server hosted in virtual instances and connected to virtual cloud storage Amazon S3.
4. Infrastructure layer includes virtual instances, virtual storage, data, network resources, and connection between them.

## **Chapter 3 Methodology**

This chapter illustrates a novel dynamic scaling methodology for improving performance of big data systems. The methodology contains three steps: splitting and modifying the system whose performance needs improvement, choosing the deployment environment, and transforming the database. The three steps are described in detail, and a guideline of how to apply these steps to enhance system performance is given at the end of the chapter.

### **3.1 Scaling the System**

A system dealing with a large amount of data in a single computer can cause troublesome performance due to processing massive transactions at the same time in a single running space. In order to respond to this problem, our approach divides the massive amount of data into smaller pieces handled in parallel computers. The challenge of scaling up the system determines how to divide the large data. Thus, the helper project algorithm is developed to scale up the system based on the amount of data to process.

#### **3.1.1 Helper Project Algorithm**

The helper project algorithm determines whether the system needs to scale up or not by applying several rules. These rules examine the data processed by the system in various aspects. Since each system has unique aspects of the data, such as the type of data used and how the data is collected, processed, and stored, the helper project algorithm is general enough to be applied to any of the systems that deal with data processing. In addition, it runs on top of the system, so it is not necessary to make many modifications in the system.

The helper project algorithm is developed to split the amount of data needed by the system to collect, process, and store, and it works based on several rules that prioritize dividing from high to low. The algorithm goes through these rules and starts splitting the data into small chunks to be processed in a timely manner. These rules consider every aspect when dividing the data into size of the data, type of data being collecting, and kind of operation and processing applied on the data. Then, each aspect is checked to see if it requires splitting. The rules considered in the helper project algorithm are listed below from high priority to low priority:

#### **3.1.1.1 Consider categories of data in splitting**

This rule is a high priority which is checked first by the algorithm and makes several separations based on the rule that will be elaborated in the following paragraph. Thus, the category of the data processed by the system needs to be known. Does the system process more than one category of data? The category of data is significant because it implies how to capture, process, and store these data, and each category has its own individual way to do so. The helper project algorithm checks if the system processes several categories of data by finding the number of categories necessary to begin the process. Then, it starts splitting the data category based on the number of categories for each chunk based on a category to be processed at a timely manner. For example, if the system has five different categories of data needed to be processed, the helper project algorithm splits the data into five small chunks. Then the system processes each small chunk in parallel instead of processing all the categories at once. Typically, the system applies several operations to process and analyze the data to gain the desired result from

these data. Consequently, the helper project algorithm checks the second rule for separation.

### **3.1.1.2 Consider analyzing data in splitting**

After the helper project algorithm examines and applies the first rule and does the necessary separation, it checks how to analyze the data. This rule regards any process of inspecting, cleansing, transforming, and modeling data with the goal of discovering useful information [3] or any operation applied on collecting data to gain desired results. Therefore, the helper project algorithm demands to find how many operations or events are applied on the collected data; based on this number, a decision is made on how to split the analysis of the data. The maximum number of operations that apply on data for separation vary from system to system. To determine the maximum number of operations the system is able to tolerate, it claims a sequence of tests on the system with a different number of operations applied. Based on the maximum number of operations performed on the data, the system executes these tasks in parallel rather than executing all operations at once.

### **3.1.1.3 Consider volume of data in splitting**

This rule is the last priority, so the helper project algorithm examines the data at the end. This rule takes into account the amount of data captured and processed when doing the splitting. As in the second rule, the maximum number of operations varies and is different from system to system, with the appropriate size of data for separation also varying. Therefore, each system requires finding the appropriate size of data that can be tolerated and then providing it to the helper project algorithm which splits the huge



amount of data into smaller amounts of data that are able to be processed. In the end, the helper project algorithm produces several tasks which have one category, maximum number of operations, and the proper amount of data that the system runs in parallel.

### **3.2 Modify the Current System**

The structure of the current system is not designed to be automated, scalable, and compatible with distributed computing. Thus, in order to apply this methodology to improve performance, a need exists to modify the current system to be compatible with distributed computing. Moreover, this methodology does not require many changes in the current system because the helper project algorithm runs on top of the current system. A modification to the current system is required to be working in the deployment environment.

Distributed computing is a model in which portions of a software system are shared among multiple computers to improve both efficiency and performance [4]. This methodology suggests two versions of the system. The first one is the base version which the system starts from and is responsible for dividing the huge amount of data by executing the helper project algorithm running on top of it. Moreover, it determines when to scale up the system based on the number of jobs the helper project algorithm produces by controlling virtual servers or machines. This version deploys the separated tasks and scales up the system into a deployment environment to work in parallel. As this version is the base and the system starts from it, it needs to be the user interface.

The second version runs in several virtual machines in the deployment environment, so it needs to be compatible with the deployment environment. Moreover, it

has major changes that modify the current system to command a line interface rather than a user interface. This version processes the small chunks produced by the helper project algorithm in parallel, and it works in the background for the base version. Also, there are some changes in storing and retrieving the data, but that varies from system to system; it is based on the structure of the system and the dependency of tasks. This methodology suggests the data, or information, is used by all distributed jobs to be in shared storage such as cloud storage, cluster, or a shared database. In this way, the distributed system avoids any redundancy or conflict while processing. By applying this approach, few modifications in the current system are needed except the two versions; one works as the interface and the second works as the background.

### **3.3 Deployment Environment**

After splitting the huge amount of data into small jobs by the helper project algorithm, the deployment environment hosts the system in several virtual machines and processes these chunks in parallel. The International Business Machines (IBM) knowledge center defines the deployment environment as a collection of configured clusters, servers, and middleware that collaborate to provide an environment to host software modules [5]. Various ways exist to use deployment environments; so it may be a network of many machines in data centers in clusters or virtual machines in cloud computing. In this approach, it is not required to apply the system in a certain environment of deployment. Many cloud computing providers exist, such as Amazon Web Services, Microsoft Azure, and Google Cloud platform. Not solely using cloud computing as the deployment environment, it could use clusters or server virtualization as the deployment environment. Moreover, the significance of the deployment environment

is to host and process the small tasks in parallel. This approach illustrates the important aspects of the deployment environment utilized to take advantage of it and to dynamically scale up the system. First, to be able to dynamically scale up the system based on the number of tasks produced by the helper project algorithm, it is necessary to control virtual servers or machines from the base version of system. The base version of the system needs the ability to start the virtual machines before the deployment in order to prepare the virtual machines for deploying and running a small task. Also, each virtual machine is responsible for shutting itself down after finishing executing assigned tasks and producing the desired result. Second, in order to begin publishing the virtual machines' version onto virtual machines in the deployment environment, it places the version of the virtual machine in an accessible place for the deployment environment. Deployment environments provide service to deploying the code, scripts, or execution file, so the base version system needs to use it to control the deployment and be eligible to initiate and verify publishing in the deployment environment. Third, if the system holds some data used as input, or produces data which is input for distributed tasks, then it demands to employ shared storage or a database in the deployment environment. Also, the helper project algorithm produces a reasonable size of tasks that depend on each other, so a need occurs to store all desired data as input in a shared place for accessibility to all virtual machines. Many deployment environments supply shared places to store various types of data such that databases, blocks, or cloud storage. Hence, the virtual machine version should be able to access shared storage to read from and write to it. Thus, it grants a solution of the redundancy and conflict in processing between dependent tasks.

Deployment environments have a significant role in this approach. This is due to several variable resources such as machines, storage, and databases based on the system needs. Therefore, it becomes able to dynamically scale up the systems based on the number of tasks produced by a helper project algorithm. Besides, it can avoid the repetition and conflict in processing between dependent tasks.

### **3.4 Database Transformation**

Big data includes a broad range of massive data produced from various sources, and the data could be structured as well as unstructured. It ranges from terabytes— $10^{12}$  bytes—to exabytes— $10^{18}$  bytes [6]. The nature of data has changed sharply in the last decade, which raises a challenge in relational databases. Moreover, systems can deal with enormous data sizes, which request rates beyond the capabilities of traditional databases. Because of the lack of a relational database in handling big data, demands for NoSQL, MongoDB, and Hadoop is increased as a solution to handle big data.

This approach focuses on improving the performance of systems dealing with huge amounts of data, but does not require these systems to use databases able to handle a high velocity of data. Moving from a traditional database into a database handling big data, it requires heavy modifications on the current system. Thus, this study suggests using the current databases but with solutions to overcome the challenges in requesting rates that are beyond the capabilities of a traditional database.

After deploying the system into several virtual machines that work in parallel, each virtual machine contains its own database, localDB, to store the results of a small task processed. In addition, when the virtual machine has accomplished processing a

small job, and stores its results to the localDB, it starts transforming records in the localDB to the centralDB, which is a shared database. The centralDB needs to be accessible for all virtual machines to query and update as needed. Besides, the virtual machines delete all records related to a processed job after moving them to the centralDB in order to prepare the localDB for the next unprocessed job. This solves the bottleneck issues caused when processing huge amounts of data. As a result, there are many distributed databases instead of the one database that cannot handle massive amounts of data.

This approach describes how to utilize the current database and become compatible working with huge amounts of data. It proposes to distribute the databases among virtual machines and then performs the transformation to a shared database between processing different jobs. After each transformation process, it deletes all data records in a distributed database in order to prepare for the next job. As a result, this overcomes the relational database challenge in handling a high velocity of data. The dynamic scaling methodology steps and requirements are shown in Figure 7.

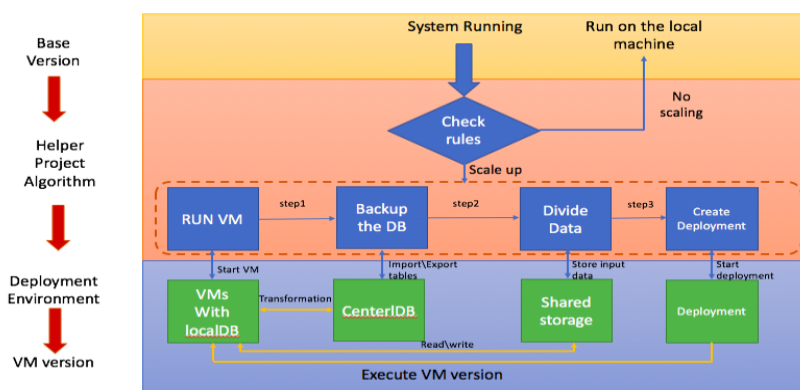


Figure 7. dynamic scaling methodology steps and requirements

### 3.5 Guidelines

The end of this chapter summarizes all the approach requirements in easy to follow and apply guidelines. These guidelines target the system facing issues in performance because of processing massive amounts of data. Also, it does not demand many modifications to the current system, and it suggests some solutions to solve this problem.

#### 1. Testing the current system.

Before applying the methodology, more about the current system needs to be known. When does performance drop down? How much data and operations can the system process in a suitable time? Does the system run different categories of data? To answer these questions requires performing a series of tests on the current system with various amounts of data, operations, and categories.

#### 2. Applying helper project algorithm.

The goal of the helper project algorithm is to separate the whole task into small tasks that the system is able to process in a suitable amount time. Thus, when performing several tests on the current system, the rules of the helper project algorithm will be known. This completely aids in applying the helper project algorithm rules in the current system. The rules of the helper project algorithm prioritize from high to low as follows:

Rule 1: divide based on the categories of data.

Rule 2: divide based on the maximum number of operations applying to the data.

Rule 3: divide based on the maximum amounts of data that the system is processing in a timely manner.

### 3. Finding deployment environment.

In order to scale up the system dynamically and to process the small tasks in parallel, there is a demand to utilize a deployment environment. The various ways to deploy the system on several servers and then finding the proper environment is significant. This methodology takes into account some important requirements in the deployment environment. Below is a list of considerations needed to be present in the deployment environment:

- Able to run and configure virtual machines remotely.
- Deploy the system onto virtual machines to run in parallel.
- Able to store shared input/output in accessible storage if necessary.
- Able for each virtual machine to connect to the local database.
- Able to set up a central database that can be accessible for all virtual machines.

Many environments provide these services and can easily implement the system in parallel. Therefore, we are able to decide what is suitable with the system's needs from the available environments such as in cloud computing: AWS, Microsoft Azure, and Google Cloud platform in the virtualization as VMware, VirtualBox, and clusters.

### 4. Transforming local database into a central database.

Each virtual machine processes a small task independently and has its own database called the localDB. To avoid the bottleneck issue in querying, it requires

moving the processed data from the local database into a central database and then resets the local database for further processing task. Consequently, there is a need for distributed databases in each virtual machine and a central database. The distributed databases work as temporary databases for processing each small job, and it is the same structure as the current database. Thus, it does not require changing the structure of the current database. Moreover, the central database must be accessible to all virtual machines to read and write. Because of that, it must be created in the deployment environment. The structure of the central database is the same as distributed databases. Thus, the list below shows what approach is required for transforming databases:

- LocalDB: distributed database located in each virtual machine.
- CentralDB: central database located in the deployment environment.



## Chapter 4 Case Study

This chapter illustrates how to apply the methodology described in Chapter 3 to the EASTWeb system. Moreover, EASTWeb faces the issue of performance due to processing massive amounts of data. The first section of this chapter describes an overview of EASTWeb as a case study to solve the problem of performance. Also, it shows the deployment environment used to scale up EASTWeb. The following section demonstrates how to perform the helper project algorithm on top of EASTWeb and how to modify EASTWeb to be compatible with applying the approach.

### 4.1 EASTWeb V2.3 overview

The EASTWeb system is set up as a stand-alone application on a single computer. When initiating the system, no complications are encountered with the performance of the system; but while in progress, the data rapidly grows which raises the failure performance issue. For example, IMERG\_Project is used as input for EASTWeb in this case study. IMERG\_Project processes 3 years of data and requires 733.468 GB which causes poor in performance when process this amount of data in the single computer. Therefore, using EASTWeb as a case study illustrates the effectiveness of the methodology and how to overcome performance failure in big data systems.

To perform the approach in EASTWeb, we need to study and test EASTWeb to define the three rules of the helper project algorithm. As a result of testing and studying, the system takes an xml file called project as an input which includes the amount of earth observation data files downloaded from online archives, different data products and its indices that apply on downloaded data. The system processes different types and sizes of

data and applies various operations on these data. These aspects are taken into account to perform the helper project algorithm.

#### **4.2 Amazon Web Service (AWS)**

In this case study, AWS is utilized to scale up and run EASTWeb in several servers in the cloud. AWS offers a set of services such as storage, database, application, and deployment. The services used to scale up EASTWeb.

AWS offers SDKs for several languages such as java, C++, Python, Ruby, and PHP. SDKs facilitate building applications to work with Amazon services. To make EASTWeb cooperate with Amazon S3, Amazon EC2, CodeDeploy, and RDB, AWS SDK for java is used to achieve scalability. A class is created for each AWS service, which includes its variables and functions. Four classes are created to utilize AWS. First, a CreateInstance class connects to ec2 instances and controls them remotely. Second, a S3 class controls the write, retrieve, and delete files from a bucket. Third, a CodeDeploy class manages and prepares the deployment of EASTWeb onto EC2 instances. Fourth, a RDS class controls connection, starting\stopping the database in the cloud (centralDB), and exporting\importing data to the centralDB. All these classes work together with EASTWeb to address the performance issues.

#### **4.3 How to apply methodology**

After testing and examining the current system, there is a clear view about the rules that the helper project algorithm considers and prioritizes. It is easy to develop the helper project algorithm by applying these rules. Also, there is a need to modify the current system to be compatible with the cloud environment. Figure 8. Shows the architecture of EASTWeb system after applying the dynamic scaling methodology.

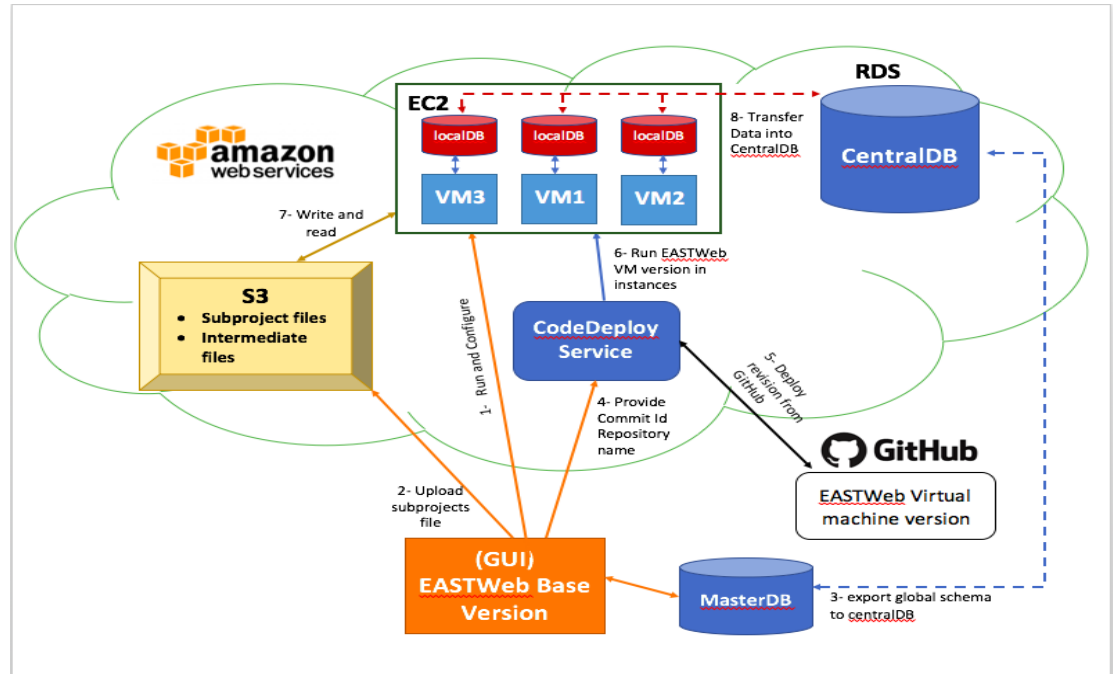


Figure 8. Overview of dynamic scaling methodology

### 4.3.1 Helper Project Algorithm

Developing the helper project algorithm to work on top of the EASTWeb system, requires defining the rules and their priorities in the case of EASTWeb. Thus, the helper project algorithm is invoked after EASTWeb runs and takes the project file as input. Based on the user-selected project entries, the helper project algorithm uses rules that prioritize from high to low as follows:

#### 4.3.1.1 Rule 1: Number of plugins considered as number of categories

Number of plugins is the highest priority for splitting the user-selected project into several subprojects. The helper project algorithm checks if a project has more than one plugin by calling the `CheckNumberOfPlugIns ()` function. This function is responsible for checking the number of plugins and making the separation based on it. Thus, if the number of plugins is more than one, then it divides the project file into

several subproject files, each with one plugin. Thereafter, it performs further splitting for each subproject based on Rules 2 and 3. This rule separates a project based on three rules. If the selected project contains a plugin, then there is no splitting and it jumps to the next rule. A plugin contains a different number of indices, so it examines the maximum number of indices that EASTWeb can process by checking the Rule 2.

```

public void CheckNumOfPlugins(){
    boolean flag;
    int index;
    int partNO = 0;
    try {
        if (NumOfPlugins > 1) {
            for (int i = 0; i < NumOfPlugins; i++) {           //divide based on the number of plugins

                for(int s=0; s < startDates.size(); s++){           //divide based on the number of years

                    noOfIndices = pluginsInfo.get(i).GetIndices().size();
                    DivideIndices = true;
                    flag = false;
                    index = 0;

                    while(DivideIndices){                       //divide based on the number of indices

                        // split the xml file project into several xml subproject files

                    }
                }
            }
        }
        else if (NumOfPlugins == 1) {
            CheckNumOfIndices();                               //jump to rule 2
        }
    } catch (ParserConfigurationException e) {
        ErrorLog.add(Config.getInstance(), "problem with creating new project.", e);
    }
}

```

#### 4.3.1.2 Rule 2: Number of indices considered as number of operations

After the helper project algorithm checks Rule 1 and finds no separation performed, it examines the number of indices as the second priority of rules. This rule invokes the CheckNumberOfIndices () function to determine the number of indices and applies the splitting. In this function, it defines the five indices as the maximum number of operations. Therefore, it produces several subproject files that have a maximum of five indices and a minimum of one index and applies additional splitting for each subproject

based on Rule 3. This rule performs two rules in dividing a project. If a project has five indices or less, there is no need for splitting and it moves to the next rule. As the least priority is the amount of data that the system processes, the helper project algorithm examines the following rule.

```

public void CheckNumOfIndices() {
    boolean flag1;
    int index;
    int partNO = 0;
    try {
        if (NumOfIndices > 5) {
            noOfIndices = pluginsInfo.get(0).GetIndices().size();

            for(int s=0; s < startDates.size(); s++){           //divide based on the number of years

                noOfIndices = pluginsInfo.get(0).GetIndices().size();
                DivideIndices = true;
                flag1 = false;
                index = 0;

                while(DivideIndices){                          //divide based on the number of indices

                    // split the xml file project into several xml subproject files
                }
            }
        }
        else if (NumOfIndices <= 5) {
            CheckNumOfYears();                                //jump to rule 3
        }
    }
    catch (ParserConfigurationException e) {
        ErrorLog.add(Config.getInstance(), "problem with creating new project.", e);
    }
}

```

#### 4.3.1.3 Rule 3: Number of years considered as size of data

The helper project algorithm checks this rule at the end, and it determines the maximum volume of data in a year of data for each subproject. EASTWeb runs a dynamic number of years since it processes the datasets from a user-selected date up to the current date. Thus, the helper project algorithm calculates the number of years from this entry by calling it the DivideYears () function. This function defines the start date and end date for each year and produces how many years in a project. It employs these dates in dividing the project. To apply the splitting of the project yearly, the

CheckNumberOfYear () function is invoked. It produces various subproject files with a year of data. If a project has less than a year of data, it is not necessary to scale up the system or divide the project.

```

public void CheckNumOfYears() {
    if (startDates.size() > 1) {

        int partNO = 0;

        for(int s=0; s < startDates.size(); s++){           //divide based on the number of years
            try {

                // split the xml file project into several xml subproject files

            }
            catch (ParserConfigurationException e) {
                ErrorLog.add(Config.getInstance(), "problem with creating new project.", e);
            }
        }
        else {
            System.out.println("No need to divide project");
            return;
        }
    }
}

```

In order to scale up the EASTWeb system based on the number of plugins, indices, and years, the helper project algorithm is developed to apply the three rules. Thus, four functions work to scatter the selected project and produce several subprojects. A description of the helper project algorithm functions is provided in Table 1.

Priority	Functions	Description
1	CheckNumberOfPlugIns ()	Check number of plugin > 1 and scatter the input file based on the number of plugins, indices, years. Otherwise, jump to next rule.
2	CheckNumberOfIndices ()	Check number of indices > 5 and scatter the input file based on the number of indices and years. Otherwise, jump to next rule.
3	CheckNumberOfYear ()	Check number of years > 1 and scatter the input file based on the number of years. Otherwise, return without splitting.
Non	DivideYears ()	Define the start date and end date for each year and number of years in selected project.

*Table 1. Functions of helper algorithm*

### 4.3.2 Modify the current system

To run EASTWeb in the cloud environment and execute the helper project algorithm on top of it, some modifications are applied to the current system as depicted in Figure 6. This approach requires two versions of EASTWeb because each version runs in a separate environment. Also, different changes are performed on each version based on the requirements of its environment. In this section, it shows the two versions of EASTWeb and what modifications are applied on each one. There are two versions, one is the base version and the other is the virtual machine (VM) version.

#### 4.3.2.1 Base version

The current system runs on a single computer as a graphical user interface (GUI), so some changes are applied to scale up the system. It is considered the base version. This

version is able to connect to AWS cloud and utilize services such as EC2, S3, and CodeDeploy. Since the base version connects to AWS cloud, it can run VM instances in the cloud in preparation for deployment by calling `StartInstance()` function. A `StartInstance()` function is accomplished connecting to AWS and starting specified ec2 instances. Moreover, this version is responsible for splitting the user-selected project by running the helper project algorithm, which produces several subproject files. These files are uploaded to cloud storage, so S3 is accessible for all VM instances by calling `WriteToS3()` function. This function creates a folder called subproject on the S3 bucket to store subproject files. In the end, it deploys the VM version to VM instances by executing the `CreateDeployment()` function. This function deploys the revision to VM instances assigned to the deployment group. Thus, before initiating the deployment, it should set up several steps which are the deployment group, creating an application, the revision location. First, it creates an application name for deployment by calling the `CreateApplication()` function. Second, it uploads the VM version to a revision location such as S3 or GitHub; in this case, the VM version is uploaded to GitHub as the revision location. Thus, to set up the revision location, a `SetRevisionLocation()` function is called. Third, the deployment group is created for successfully deploying by invoking `CreateDeploymentGroup()` function.

The EASTWeb base version runs on a single computer as the user interface and is responsible for executing the helper project algorithm. The algorithm decides if a need exists to scale up the system or not. In the case of scaling up the EASTWeb, the resources in the cloud should be prepared for deployment and run the VM version of EASTWeb on several VM instances in the background.



#### 4.3.2.2 Virtual machine version

The virtual machine version is extracted from the current system and runs on several VM instances on the cloud. To be compatible with the cloud environment, several changes are performed on this version. Since the current system is a graphical user interface GUI, there is a need to modify it to the command line interface CLI. The reasons for using CLI is this version executes on a variety of servers as a background to the base version. These servers do not require interaction with users. Moreover, it must be lightweight for greater efficiency and speed.

This version starts executing on a virtual machine instance after the base version creates the deployment. It takes a subproject file as input from the cloud storage S3. These subproject files are already produced by the helper project algorithm and written to S3 in the base version. As a subproject file is downloaded into a virtual machine instance, it is deleting from the cloud storage to avoid conflict between virtual machine instances. Therefore, each virtual machine instance in the cloud is able to download a subproject file and process it through EASTWeb steps. When the VM version is terminated processing the subproject files and storing its related results in the database, it rechecks the cloud storage for any unprocessed subproject files. If an unprocessed file is located, it does the same steps again. Otherwise, the system shuts down automatically as well as the VM instance.

The major processing steps in EASTWeb generate several intermediate files stored in the local drive. In order to avoid the redundancy in processing steps, each virtual machine uploads the intermediate files that processing steps are produced to cloud storage. However, there are certain steps that upload its files to cloud storage such as the

downloading and indices steps. This is due to the height of dependency on these two steps. The download step produces stream files from online earth observation archives and stores them locally for further processing [1]. If several virtual machines are processing the same year of data, each one re-downloads the same files and stores them locally, which causes repetitions in the download step. Thus, in the downloading step uploads, produced files into cloud storage is by invoking the WriteFilesToS3() function. Moreover, the indices step calculates various indices for each data product and generates files in GeoTIFF format [1]. A data product has a cumulative index. This index needs to check the previous year to calculate it. Therefore, it requires checking the indices in each data product whether it has a cumulative index or not. In the case of a cumulative index, the related index files are uploaded to cloud storage by calling the WriteFilesToS3() function. Otherwise, it does not upload any index files to the cloud storage. As a result, the index files are accessible for varied virtual machines that process the next year of a cumulative index.

Two versions of EASTWeb work together to scale up the system based on helper project algorithm rules. The base version runs as an interface of the system, and the VM version works in the background of the base version. Workflow of the EASTWeb two versions is illustrated in Figure 9.

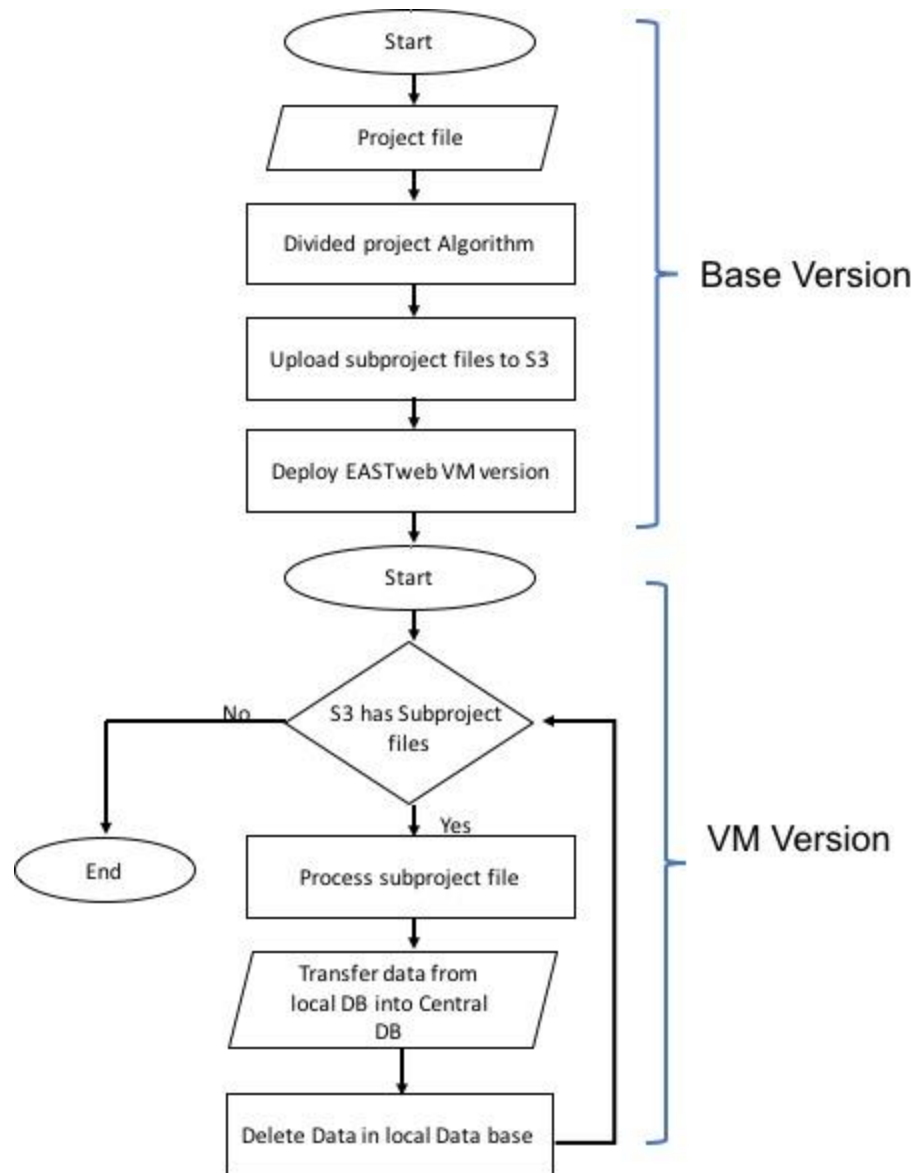


Figure 9. The workflow of two versions of EASTWeb

### 4.3.3 Database Transformation

EASTWeb utilizes the PostgreSQL database to store and manipulate the outcomes of processing data [1]. PostgreSQL is a traditional relational database (RDBMS), which has limitations storing and retrieving huge amounts of data. Using RDBMS in handling big data causes delays in storing and retrieving data. For example, EASTWeb starts

facing delays in database operations after processing three years of data because it produces a massive amount of records in processing a year of data for some data products. However, RDBMS can be used to handle and store big data by applying the mechanism of transforming the database yearly to a central database.

To perform this approach in the EASTWeb VM version, Amazon Relational Database Service (RDS) is used to provide a database over the cloud with several database engines such that PostgreSQL, Oracle, MySQL, and others. By using this service, the PostgreSQL database is created to be a central database (centralDB). This database is accessible for all virtual machine instances to store and retrieve data. It works as a place to accumulate the results of processing data in each virtual machine instance. Therefore, each virtual instance has its own database as a local database (localDB). A localDB works independently along with EASTWeb VM version to store and manipulate the outcomes of processing subproject data. When the EASTWeb VM version terminates all processing steps and stores the data to the localDB, it starts moving the localDB into the centralDB by several steps. These steps work with an assumption of conflict of the primary keys between tables while in transformation. Thus, there is a need to know the tables that may occur in conflict. In the EASTWeb case, a global schema has six tables and their primary keys may overlap. The steps are listed below:

- Step 1: Export the six tables having conflict in primary keys from the master database related to the EASTWeb base version to make a backup of the last update of these tables by calling the `CreatePG_DumpForGlobalSchema(scemaName, tableName)` function. Then, import these tables to the centralDB to be up to date by invoking the

ImportPGdumpFile(fileName) function. This step is done before creating the deployment by the EASTWeb base version. Also, it is significant to make the primary key of these tables concurrent.

- Step 2: In the VM version, the global schema in the centralDB is shared among all virtual machines. While each VM processes a subproject, the global schema is updated with new records. This is important to avoid the conflict of primary keys and foreign keys.
- Step 3: After a subproject is processed and stores its result in a subproject schema in the localDB, the subproject schema is exported to the centralDB. After exporting each subproject schema, it evacuates the localDB. This is important to prepare the localDB for unprocessed sub-projects and avoid delays in database operations.
- Step 4: Copy the six tables of the global schema from the centralDB after complete processing of all subproject files. Then export all subproject schemas to the master database. This step is done by the EASTWeb base version. This step updates the master database for further processing.

This approach tackles the issue of handling huge amounts of data in traditional databases. Also, it provides a solution for overlapping the primary keys of relational tables by performing these steps.

## Chapter 5 Evaluation

This chapter illustrates evaluation of the dynamic scaling methodology. Dynamic scaling methodology is utilized as a helper project algorithm and deployment environment to enhance the performance of big data systems. Consequently, the first section shows how dynamic scaling methodology is evaluated by applying it in the case study, along with the performance results. The following section presents an evaluation in utilizing and analyzing the deployment environment in two aspects: simplicity and flexibility. At the end of this chapter, we show the guidelines are appropriate to follow by evaluating several criteria.

### 5.1 How does the dynamic scaling methodology work?

In order to show the efficiency of dynamic scaling methodology in improving the performance of big data systems, it is applied to the EASTWeb system as a case study. The case study provides an overview of how our methodology works with systems dealing with big data. Moreover, the results of applying the dynamic scaling methodology are collected by comparing the performance of EASTWeb with and without the dynamic scaling approach. These results are considered a measurement of the effectiveness of the dynamic scaling methodology in improving the performance of systems handling enormous amounts of data.

#### 5.1.1 Applying dynamic scaling methodology in the case study

To prove the dynamic scaling approach overcomes the performance failure in systems processing huge volumes of data, a need exists to implement our approach in the case study. Thus, the issue for the EASTWeb system, which is used as a case study, is the performance of processing massive amounts of data. For example, the EASTWeb system

spends a minimum of one day to process one year of data. It is time consuming when processing more than one year of data, and it applies many calculations on the data. In additions, some projects are not able to be processed by EASTWeb because of its demands in processing massive amounts of data. Therefore, the EASTWeb system is an appropriate case study to apply our methodology. Moreover, the success of the dynamic scaling methodology is measured by checking the amount of modifications performed on the current system and the effectiveness of the helper project algorithm in dividing and scaling after applying it to EASTWeb.

The goal of the dynamic scaling methodology is improving performance without applying many modifications to the current system. This is achieved after applying our methodology on EASTWeb. Since the helper project algorithm runs on top of the EASTWeb system, it divides the amounts of tasks based on its rules without affecting the functionality of the current system. Also, to scale up the system and make it compatible with the deployment environment, the command line interface (CLI) is extracted from the current system. CLI runs on several virtual machines; therefore, it is modified to read/write from cloud storage and store the results in the cloud database. As a result, the dynamic scaling methodology does not demand a redesign of the current system.

In addition, applying dynamic scaling methodology in EASTWeb examines the efficiency of the helper project algorithm in dividing the amounts of data based on its rules. Also, it checks if the helper project is properly scaling up/down the system. To achieve that, EASTWeb runs various sizes of projects; some are needed for scaling up the system to be processed and others are not. The helper project algorithm divides the project meeting the rules and then properly scales up the system. If a project does not

need to be divided, the helper project runs the project on the local computer. Results of performance of EASTWeb after applying our approach is described in the following section.

### **5.1.2 Performance comparison of case study**

To prove the dynamic scaling methodology is effective in improving the performance of big data systems, we compare the performance of EASTWeb with and without our methodology. Two scenarios are performed on EASTWeb; then we compare the results of each scenario. Each scenario runs a project called the IMERG\_Project. This project has several entries such as a plugin, an index, and three years of data. The first scenario is the dynamic scaling methodology is run on the top of EASTWeb. Thus, the helper project algorithm divides this project into three subprojects based on its rules. Each subproject runs on a virtual machine to work in parallel. The second scenario is EASTWeb runs without our methodology, so the IMERG\_Project is not separated into several subprojects. Therefore, the entire project runs in a sequence. The results of performance for each scenario is shown in Table.2.



<b>Scenarios</b>	Scenario1: EASTWeb with the dynamic scaling methodology	Scenario2: EASTWeb without the dynamic scaling methodology
<b>Project</b>	IMERG_Project	IMERG_Project
<b>Description</b>	Three subprojects run on several virtual machines in parallel.	A project run on the single commuter.
<b>Performance</b>	Takes 13 hours to process 3 years of data	Takes 23 hours and 30 minutes to process 3 years of data

*Table 2. the performance results of two EASTWeb Scenarios*

The dynamic scaling methodology is able to decrease the running time to 10 hours as shown in scenario1. Since our methodology divides and runs the IMERG\_Project on several virtual machines working in parallel, the performance of EASTWeb is improved significantly. As a result, the dynamic scaling methodology illustrates its effectiveness in enhancing the performance of systems processing huge amounts of data.

## **5.2 How does deployment approach work?**

One of the dynamic scaling methodology components is the deployment environment. Many deployment environments exist: cloud, cluster, virtualization, or grid. Cloud computing is utilized as a deployment environment to scale up the system and employ its services in scaling. Moreover, several providers of cloud computing exist, such as Google Cloud platform, Microsoft Azure, Amazon Web Service, and IBM. Therefore, the deployment approach takes into account the simplicity and flexibility in the evaluation and choice of the deployment environment. Simplicity and flexibility are two aspects needed in the deployment environment to fulfill the success of our methodology.

### **5.2.1 Simplicity**

To satisfy the simplicity of the deployment environment, the amazon web service (AWS) is utilized as a deployment environment to scale up/down the systems. AWS provides a wide range of services that can be employed to scale up the system without much effort. Offering a variety of services helps to meet all the system's needs, which allows choosing the operating system, web application platform, database engines, programming languages, and other services [17]. For example, AWS has six instance families and 38 instance types with several regions [19]. This offers more options for clients in the computing service. All services are able to integrate and communicate together to address the scaling issue. Moreover, AWS provides software developer kits (SDKs) in various languages, which simplify using AWS services in applications. Consequently, using AWS achieves simplicity in the deployment approach. However, the

dynamic scaling approach is not restricted to AWS, and it could work along with any cloud platform.

### **5.2.2 Flexibility**

One concern in the dynamic scaling methodology and the deployment approach is flexibility. Flexibility is evaluated on both the dynamic scaling methodology and deployment approach. To accomplish the flexibility in the dynamic scaling methodology does not require using a specific deployment environment; it is flexible to work with any environment. Thus, our methodology provides the requirements that a system needs to scale it up such as virtual machines, databases, and storage in the deployment environment. These requirements are mandatory for success of the deployment, not the deployment environment. Therefore, our methodology does not restrict using a certain environment, so it fulfills the flexibility in the dynamic scaling methodology.

This deployment approach should to be flexible to gain successful deployment of the application. Moreover, flexibility in the deployment environment speeds up communication between services necessary for scaling up the system. Therefore, AWS is selected as a deployment environment. Flexibility is one of the significant features in AWS. The services work and communicate together with the application to automatically decide appropriately in dealing with requests. [18]. AWS enables configuration of the virtual machines using either a pre-configuration or custom machine image (AMIs). In storage, it provides a temporary block and object storages. AWS fully supports relational NoSQL and big data databases. All characteristics of AWS are achieved through flexibility of the deployment approach.

### **5.3 Guideline**

In Chapter 3 we provide guidelines for systems facing a dilemma of failures in performance in processing huge amounts of data. To ensure the guidelines are appropriate to follow, they are evaluated on several criteria such as simplicity, description, and information. Therefore, to achieve simplicity in the guideline, it is written as an easy to follow list. Also, it is shown as a diagram to facilitate understanding. To make each step more descriptive and more understandable, the guideline contains a brief description. This fulfills the descriptive criteria of the guidelines. For achieving the informative criteria, each guideline shows several examples and background information, which gives the reader a general overview.

### **5.4 Compare our work to others works**

#### **5.4.1 Dynamic Scalability of web applications on a virtualized Cloud computing environment [10]**

This research provided a novel scenario for dynamic scaling of a web application by employing scaling indicators. The scaling indicators, related to the web application, were number of concurrent users, number of active connections, average response times per request, and other indicators. On the other hand, this thesis illustrated a methodology for dynamic scaling of big data systems by using three rules. The rules used as indicators and specified by each system applied dynamic scaling to improve performance.

#### **5.4.2 ECG monitoring and analysis system [9]**

This research provided an overview of how to design a dynamic scalable system. Thus, the helper project algorithm developed takes advantage in its results and system design. Unlike the helper project algorithm, this approach focuses on scaling up the system based on two parameters, user requests and response time, to enhance the quality of services of the web application. The helper project algorithm dynamically scales up the system based on three rules with various priorities. Also, it is flexible to scale the system at any deployment environment.

#### **5.4.3 High performance analytics of big data with dynamic and optimized Hadoop cluster [16].**

This research presented a model to handle analyzing a large amount of data on a cloud environment and matched the requirements of safety, easily scalable, and high efficiency. Unlike our methodology, they used a Hadoop tool to schedule jobs and MapReduce to distribute tasks through clusters. Moreover, they used virtual machines to host the application services and database.

## Chapter 6 Conclusion

Dynamic scaling methodology is designed to tackle performance failures in big data systems. This chapter contains two sections. The first section demonstrates a summary of the thesis with an overview of our work, and the second section discusses future work in areas of dynamic massive-scaling researches.

### 6.1 Conclusion

A novel dynamic scaling methodology is developed in this research to deal with the performance failure of systems that process huge amounts of data. This methodology involves a novel algorithm called the helper project to divide a task into several smaller tasks based on various aspects in a big data perspective. These aspects known as algorithm rules are prioritized from high to low for dividing the project. Also, the helper project decides whether it is necessary to scale up, or not, the system based on the number of tasks produced after separation. Moreover, our methodology provides an overview of how to choose the deployment environment and what services are needed for scaling systems. The most significant features of the dynamic scaling methodology are that they do not restrict in certain environments.

The primary interest in this methodology is to apply it without doing many modifications to the current system. Moreover, it addresses the bottlenecks of relational databases during processing huge amounts of data by suggesting a database transformation approach.

The dynamic scaling methodology significantly improves the performance of systems handling massive amounts of data. It divides the task into several smaller tasks by applying the helper project algorithm. Also, it applies the necessary scaling up of the system to run in several virtual machines in the deployment environment. The methodology is applied to the EASTWeb system, which suffers a severe performance issue in processing huge amount of data. We compared the speed of running the same project in EASTWeb with and without dynamic scaling methodology, and results showed that applying the dynamic scaling methodology significantly improved the performance.

## **6.2 Future Work**

The dynamic scaling methodology improves the performance of big data system using cloud computing. As a direction of future work, the dynamic scaling methodology will work along with other deployment environment to be more flexible and general. Moreover, in order to enhance the efficiency of the dynamic scaling methodology, a new algorithm will be developed to scale the systems in two directions horizontally and vertically.

### **6.2.1 Consider other deployment environment**

There are several deployment environments such as cluster and virtualization [21]. Therefore, utilizing the dynamic scaling methodology in other deployment environments will create a more general and flexible methodology. Using virtualization

in VMware [22] along with the dynamic scaling methodology will be the next step to enhance our work.

### **6.2.2 Consider the vertical scaling**

Our methodology focuses on horizontal scaling to improve the performance of big data systems. Horizontal scaling capability increases the resources, such as hardware or software, to improve performance [20]. The helper project is able to increase and decrease the virtual machine instances based on the number of tasks to be processed by the system. Thus, to improve the dynamic scaling methodology, an algorithm will be developed to manage the scaling vertically. The vertical scaling ability increases resources such as increasing memory space and CPU to a machine [20]. To apply the vertical scaling to our methodology, a need exists to identify the proper resources for each task. Applying scaling in both directions, horizontally and vertically, will provide better enhancement in the performance of big data systems.



## LITERATURE CITED

- [1] Liu, Y., M. D. Devos, M. Abdul-Rahim, J. Hu, and M. C. Wimberly. 2016. EASTWeb framework - a plug-in framework for constructing geospatial health applications. In: 2016 IEEE International Conference on Electro-Information Technology (EIT). Grand Forks, ND: 0627-0632.
- [2] Liu, Y., J. Hu, I. Snell-Feikema, M. S. VanBemmel, A. Lamsal, M. C. Wimberly. 2015. Software to Facilitate Remote Sensing Data Access for Disease Early Warning Systems. *Environmental Modeling and Software*. Vol. 74, p. 247-257.
- [3] Bihani, Prateek, and S. T. Patil. "A comparative study of data analysis techniques." *International Journal of Emerging Trends & Technology in Computer Science* 3, no. 2 (2014): 95-101.
- [4] distributed computing: <http://whatis.techtarget.com/definition/distributed-computing> (accessed December 15, 2017).
- [5] Deployment environments: [https://www.ibm.com/support/knowledgecenter/en/SSTLXK\\_7.5.1/com.ibm.wbpm.ref.doc/help\\_nd/index.html](https://www.ibm.com/support/knowledgecenter/en/SSTLXK_7.5.1/com.ibm.wbpm.ref.doc/help_nd/index.html) (accessed December 15, 2017).
- [6] Why traditional database systems fail to support "big data": <http://marketrealist.com/2014/07/traditional-database-systems-fail-support-big-data/> (accessed December 15, 2017).
- [7] Hashem, Ibrahim Abaker Targio, Ibrar Yaqoob, Nor Badrul Anuar, Salimah Mokhtar, Abdullah Gani, and Samee Ullah Khan. "The rise of "big data" on cloud computing: Review and open research issues." *Information Systems* 47 (2015): 98-115.
- [8] Amazon Web Service: <https://aws.amazon.com>
- [9] Pandey, Suraj, William Voorsluys, Sheng Niu, Ahsan Khandoker, and Rajkumar Buyya. "An autonomic cloud environment for hosting ECG data analysis services." *Future Generation Computer Systems* 28, no. 1 (2012): 147-154.
- [10] Chieu, Trieu C., Ajay Mohindra, Alexei A. Karve, and Alla Segal. "Dynamic scaling of web applications in a virtualized cloud computing environment." In *E-Business Engineering, 2009. ICEBE'09. IEEE International Conference on*, pp. 281-286. IEEE, 2009.
- [11] Ostermann, Simon, Alexandria Iosup, Nezih Yigitbasi, Radu Prodan, Thomas Fahringer, and Dick Epema. "A performance analysis of EC2 cloud computing services for scientific computing." In *International Conference on Cloud Computing*, pp. 115-131. Springer, Berlin, Heidelberg, 2009.
- [12] what is AWS EC2? : <https://www.sumologic.com/aws/what-is-aws-ec2/> (accessed December 15, 2017).

- [13] Amazon Simple Storage Service (Amazon S3): <http://searchaws.techtarget.com/definition/Amazon-Simple-Storage-Service-Amazon-S3> (accessed December 15, 2017).
- [14] AWS CodeDeploy (Amazon Web Services CodeDeploy): <http://searchitoperations.techtarget.com/definition/AWS-CodeDeploy-Amazon-Web-Services-CodeDeploy> (accessed November 10, 2017).
- [15] Li, Yang, Li Guo, and Yike Guo. "An efficient and performance-aware big data storage system." In International Conference on Cloud Computing and Services Science, pp. 102-116. Springer, Cham, 2012.
- [16] Pradhananga, Yanish, Shridevi Karande, and Chandraprakash Karande. "High performance analytics of bigdata with dynamic and optimized hadoop cluster." In Advanced Communication Control and Computing Technologies (ICACCCT), 2016 International Conference on, pp. 715-720. IEEE, 2016.
- [17] AWS vs Azure vs GCP: <http://www.globaldots.com/aws-vs-azure-vs-gcp-ups-downs-public-cloud-trifecta/> (accessed December 15, 2017).
- [18] 6 reasons why we chose AWS: <http://aptuz.com/blog/6-reasons-why-we-chose-aws/> (accessed December 15, 2017).
- [19] AWS vs Azure vs Google: <https://cloudacademy.com/blog/public-cloud-war-aws-vs-azure-vs-google/> (accessed December 15, 2017).
- [20] horizontal scalability: <http://searchcio.techtarget.com/definition/horizontal-scalability> (accessed December 15, 2017).
- [21] Deployment environments: [https://www.ibm.com/support/knowledgecenter/SSFPJS\\_8.5.0/com.ibm.wbpm.ref.doc/help\\_nd/index.html](https://www.ibm.com/support/knowledgecenter/SSFPJS_8.5.0/com.ibm.wbpm.ref.doc/help_nd/index.html) (accessed December 15, 2017).
- [22] VMware: <https://www.vmware.com>
- [23] IBM: <https://www.ibm.com/us-en/>
- [24] Google Cloud Platform: <https://cloud.google.com>
- [25] Microsoft Azure: <https://azure.microsoft.com/en-us/>